بسم الله الرحمن الرحيم

لغة البرمجة جافا

Java Programming Language

الدرس الثاني :
الترجمة و العمليات في جافا

أول برنامج جافا :

البرنامج يقوم بطباعة سلسلة و من ثم يقوم بطبع التاريخ و ذلك اعتمادا على صف Date الموجود في مكتبة جافا المعيارية

```
import java.util.*;

public class HelloDate {

public static void main(String[] args) {

 System.out.println("Hello, it's: ");

 System.out.println(new Date());

 }
}
```

شرح على الكود :

- في بداية ملف أي برنامج نقوم ببناءه نضع عبارة `import` و ذلك لاحضار أي صفوف إضافية سنستخدمها في الكود

هناك مكتبة من الصفوف تكون معرفة بشكل أوتوماتيكي في كل برامج جافا و هي المكتبة `java.lang`

الصف `System` موجود في مكتبة `java.lang` لذلك يمكن استخدامه مباشرة

أحد حقول الصف `System` هو الحقل `out` و هو حقل `static`

الحقل `out` غرض من نوع `PrintStream` . يوجد في الصف `PrintStream` عدة طرق يمكن استدعائها بواسطة الغرض `out` .. ما يهمنا الآن هو الطريقة `println ()` أو `print ()` و تقوم كل من الطريقتين بطباعة العبارة الممرة لها كوسيط و الخلاف الوحيد هي أنه `println ()` يطبع العبارة و ينزل سطر جديد .
لذلك في أي برنامج جافا إذا أردنا طباعة أي عبارة نكتب احدى التعليمتين :

```
System.out.println("any thing");

System.out.print("any thing");
```

- اسم الصف مطابق تماما لاسم الملف . عندما نقوم بكتابة برنامج مؤلف من عدة صفوف موجودة في نفس الملف يجب أن يكون اسم أحد هذه الصفوف مطابق لاسم الملف و هذا الصف يجب أن يحتوي التابع main()
- التابع main() له الشكل التالي :

```
public static void main ( String [] args ) {

}
```

وسيط تابع الـ ()main عبارة عن مصفوفة من السلاسل .. لم نستخدم هذه المصفوفة في برنامجنا و لكن المترجم يستخدم هذه المصفوفة عند تنفيذ البرنامج في محرر الاوامر في حال أردنا تمرير بارمترات للبرنامج

- السطر التالي :

```
System.out.println(new Date());
```

تم إنشاء غرض من نوع Date و مرر إلى التابع () println . لن نكون بحاجة للغرض Date بعد انتهاء عملية الطباعة . يقوم الـ garbage collector بأخذ هذا الغرض و حذفه من الذاكرة

<u>الترجمة و تنفيذ البرنامج :</u>

لتنفيذ البرنامج السابق نستخدم برنامج يسمى JDK و هو برنامج يحتوي على الأدوات الضرورية لكتابة و فحص و تنفيذ برامج جافا .

نستخدم الأداة javac لترجمة البرنامج كما يلي :

Javac HelloDate.java

عند نجاح تنفيذ السطر السابق نحصل على ملف من نوع class. و اسمه يطابق اسم ملف البرنامج .. أي HelloDate.class

لتنفيذ البرنامج نكتب ما يلي :

Java HelloDate

يقوم السطر التالي بتفسير الملف HelloDate.class و يقوم بطباعة العبارة و الوقت الحالي

<u>ترجمة و تفسير :</u>

لدينا ثلاث مراحل للحصول على برنامج جافا ينفذ بشكل صحيح :

١- كتابة كود المصدر للبرنامج و يكون الملف من نوع java.
٢- ترجمة البرنامج باستخدام javac ... هذه العملية تعطينا ما يسمى ملف bytecode
Java bytecode: عبارة عن ملف تعليمات لمعالج جافا و هو يتولد كنتيجة لترجمة برنامج جافا و نوع النلف الناتج class.
إن نتيجة ترجمة الملف java. هي نفسها في حال لو قمنا بترجمة الملف على جهاز آخر ... أي أن ملف class.سيتولد نفسه بعض النظر عن الجهاز الذي تمت عليه الترجمة (حتى لو كان ماكنتوش)
٣- تفسير ملف الـ bytecode عن طريق معالج الجافا .. و نتيجة التفسير تعطي النتيجة
و لكن لا يوجد معالج جافا كـ هاردوير .. لذلك هذا المعالج يضاف بشكل برمجي أي أن هناك برنامج يقوم بأخذ ملف الـ bytecode و يقوم بتفسيره .. كل جهاز لديه مفسر جافا مختلف عن الآخر و ذلك حسب النظام المستخدم
و هذا ما يسمى java Virtual Machine .. مفسر جافا يعمل على نظام حاسوب

التعليقات في برامج جافا :

توضع التعليقات في برامج جافا بعد (//) ... في حال سطر واحد ، و إذا كان لدينا عدة أسطر يمكن وضعها كتعليق بين (/*.....*/)

التحكم في تدفق البرنامج :

في جافا نحن نعالج البيانات و الأغراض و نضنع القرارات عن طريق تعليمات التحكم .. ، جميع هذه تعليمات التحكم في جافا تشابه تعليمات التحكم في C و ++C ، و لكن جافا أضافت بعض التفضيلات و التخصيصات على هذه التعليمات ...

استخدام العمليات في جافا :

العملية لها معامل واحد أو أكثر و تنتج قيمة جديدة ، استدعاء العمليات على معاملاتها يختلف عن استدعاء طريقة مع معاملاتها و لكن التأثير نفسه .

الأولويات :

أولوية العملية تعرف كيفية تقييم تعبير بعدة عمليات ، جافا لديها قواعد محددة لترتيب التقييم ، هذه القواعد مشابهة لما تعلمناه في لغة ++C ...

عملية الاسناد :

عملية الاسناد تنجز بواسطة العملية (=) .. و تعني خذ القيمة الموجودة في الطرف اليميني و اسندها للمتحول في الطرف اليميني .،

القيمة اليمينية يمكن أن تكون : ثابت – متحول – تعبير ، و لكن القيمة اليسارية يجب أن تكون حصرا عبارة عن متحول

مثال :

```
A = 4 ;      // ........ true
4 = A ;      // ...... false
```

إن اسناد قيمة إلى متحول أولي (primitive) تعني نسخ القيمة إلى المتحول و ذلك لأن المتحول الأولي يحتوي على القيمة بشكل مباشر ..

عندما اسناد أغراض هناك بعض الاختلافات : عندما نسند غرض إلى آخر فهذا يعني نسخ العنوان من مكان إلى آخر أي عندما نقول C = D فإننا تنفيذ هذه العملية يؤدي إلى أن كلا من C و D أصبحا يؤشران على نفس الغرض

مثال :

```
public class Number {

 int i;

 public static void main(String[] args) {

  Number n1 = new Number();
  Number n2 = new Number();
  n1.i = 9;
  n2.i = 47;
  System.out.println("1: n1.i: " + n1.i +
  ", n2.i: " + n2.i);
  n1 = n2;
  System.out.println("2: n1.i: " + n1.i +
  ", n2.i: " + n2.i);
  n1.i = 27;
  System.out.println("3: n1.i: " + n1.i +
  ", n2.i: " + n2.i);
 }
}
```

هذا المثال يوضح تماما فكرة إسناد المتحولات ..

بعد أن قمنا باسناد n2 إلى n1 .. فإن تعديل أي منهما سيرى للآخر و ذلك لأن كلاهما يؤشران إلى نفس الغرض ..

و كذلك إذا قمنا بتمرير غرض ما إلى تابع و عدلنا عليه فإن التغيير سيبقى و ذلك لأن التعديل هو على الغرض الفعلي و ليس على القيمة

مثال :

```
public class PassObject {
char c;
static void f(Letter y) {
y.c = 'z';
}
public static void main(String[] args) {
PassObject x = new PassObject ();
x.c = 'a';
System.out.println("1: x.c: " + x.c);
f(x);
System.out.println("2: x.c: " + x.c);
}
}
```

العمليات الرياضية :

العمليات الرياضية في لغة جافا هي نفسها الموجودة في بقية اللغات : الجمع (+) ، الطرح (-) ، الضرب (*) ، القسمة (/) ، باقي القسمة (%)

و كذلك عملية الاسناد مع الجمع (=+) ، و العمليات المماثلة ...

مثال :

```
import java.util.*;

public class MathOps {

static void prt(String s) {
System.out.println(s);
}

static void pInt(String s, int i) {
prt(s + " = " + i);
}

static void pFlt(String s, float f) {
prt(s + " = " + f);
}

public static void main(String[] args) {

Random rand = new Random();
int i, j, k;
j = rand.nextInt() % 100;
k = rand.nextInt() % 100;
pInt("j",j); pInt("k",k);
i = j + k; pInt("j + k", i);
i = j - k; pInt("j - k", i);
i = k / j; pInt("k / j", i);
i = k * j; pInt("k * j", i);
i = k % j; pInt("k % j", i);
j %= k; pInt("j %= k", j);

float u,v,w; // applies to doubles, too
v = rand.nextFloat();
w = rand.nextFloat();
pFlt("v", v); pFlt("w", w);
u = v + w; pFlt("v + w", u);
u = v - w; pFlt("v - w", u);
u = v * w; pFlt("v * w", u);
u = v / w; pFlt("v / w", u);

u += v; pFlt("u += v", u);
u -= v; pFlt("u -= v", u);
u *= v; pFlt("u *= v", u);
u /= v; pFlt("u /= v", u);
}
}
```

تم بحمد الله